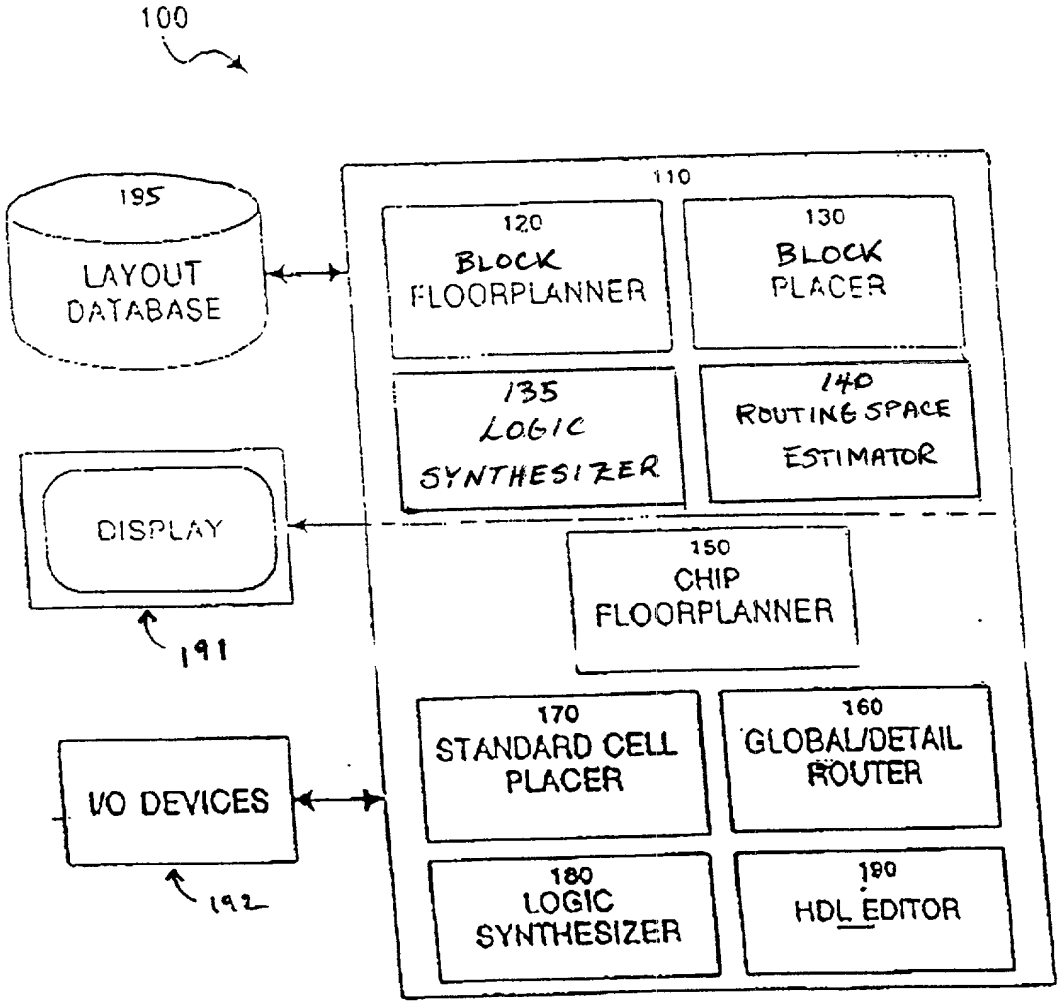
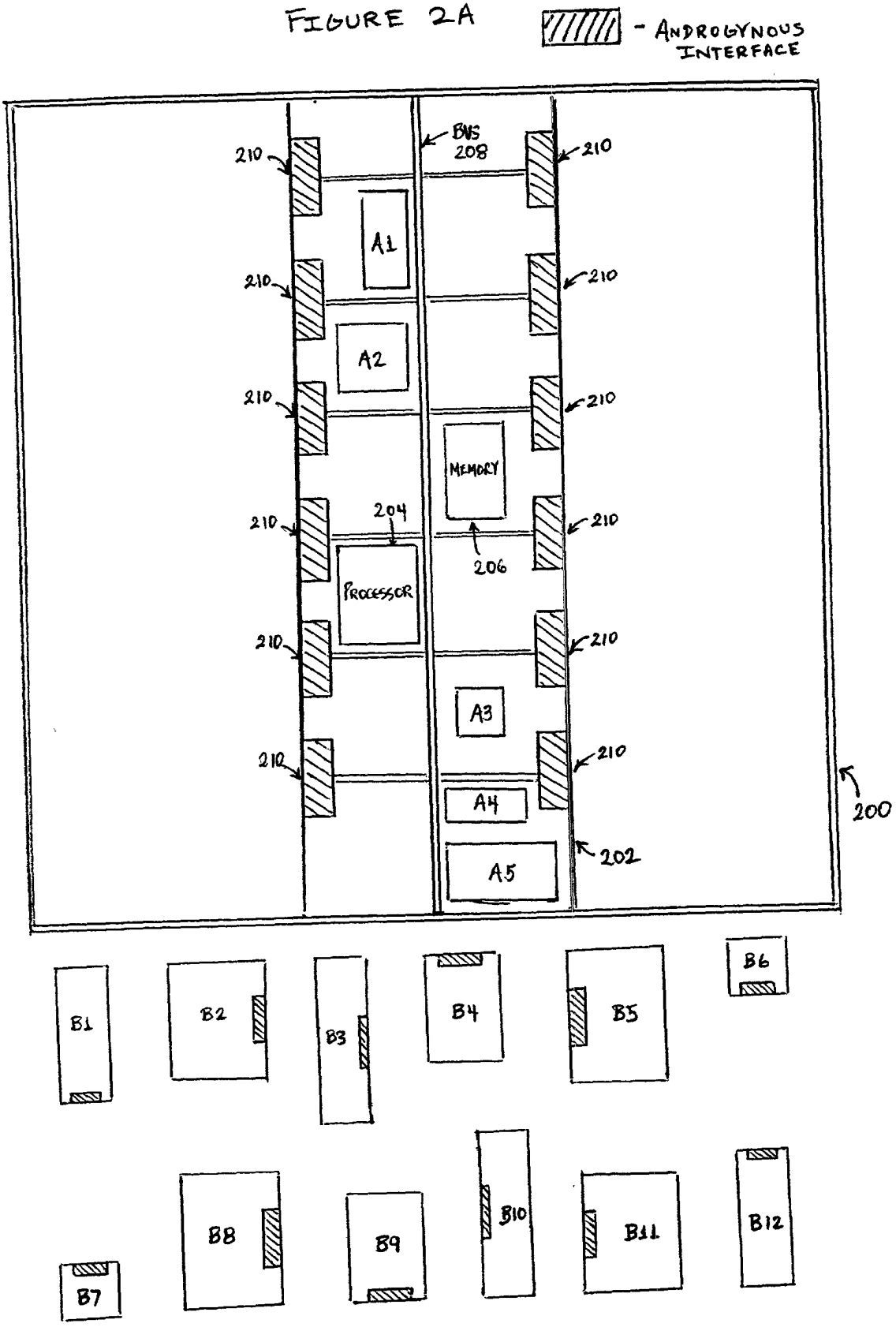

100
110
195
LAYOUT DATABASE
120
BLOCK FLOORPLANNER
130
BLOCK PLACER
135
LOGIC SYNTHESIZER
140
ROUTING SPACE ESTIMATOR
DISPLAY
191
150
CHIP FLOORPLANNER
I/O DEVICES
192
170
STANDARD CELL PLACER
160
GLOBAL/DETAIL ROUTER
180
LOGIC SYNTHESIZER
190
HDL EDITOR

FIGURE 1

FIGURE 2A


- ANDROGYNOUS INTERFACE
BUS 208
210
A1
A2
A3
A4
A5
MEMORY
204
206
PROCESSOR
200
202
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
B12

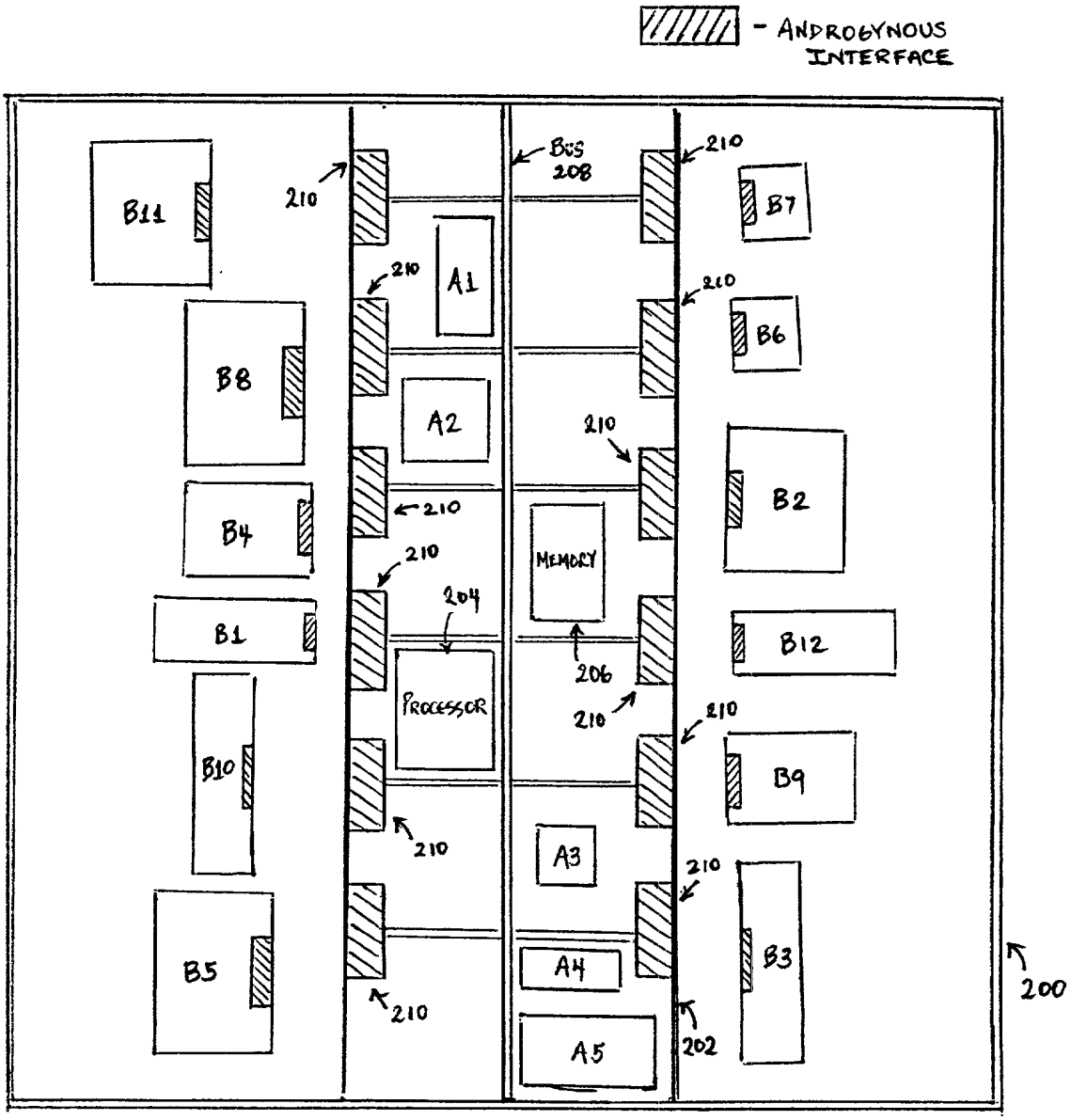

- ANDROGYNOUS INTERFACE
B11
210
BUS 208
210
B7
A1
210
210
B6
B8
A2
210
B2
B4
210
MEMORY
210
204
B1
B12
PROCESSOR
206
210
210
B9
B10
210
A3
210
B3
B5
A4
210
200
202
A5

FIGURE 2B

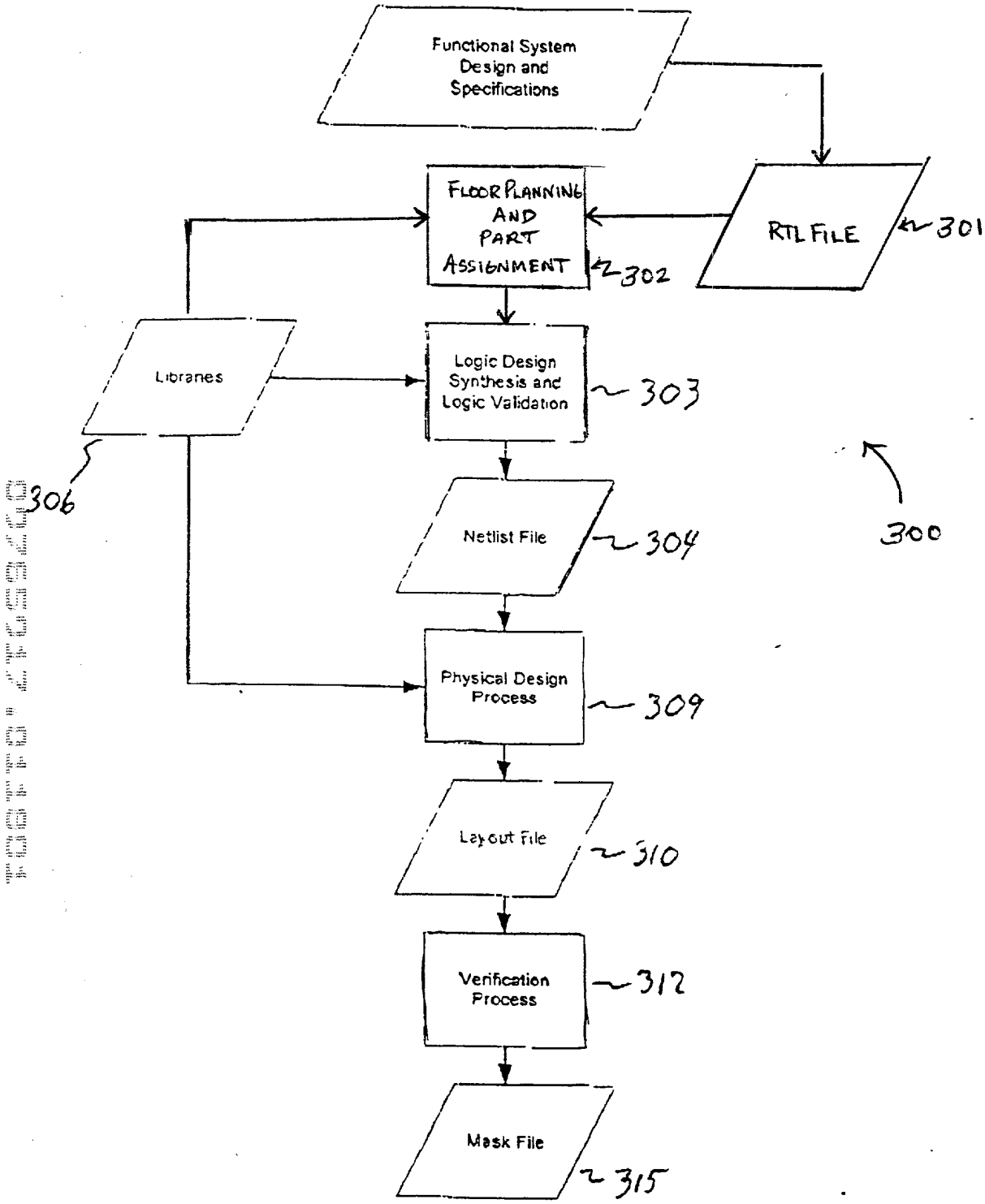

Functional System Design and Specifications
FLOOR PLANNING AND PART ASSIGNMENT
302
RTL FILE
301
Libraries
306
Logic Design Synthesis and Logic Validation
303
Netlist File
304
300
Physical Design Process
309
Layout File
310
Verification Process
312
Mask File
315

FIGURE 3

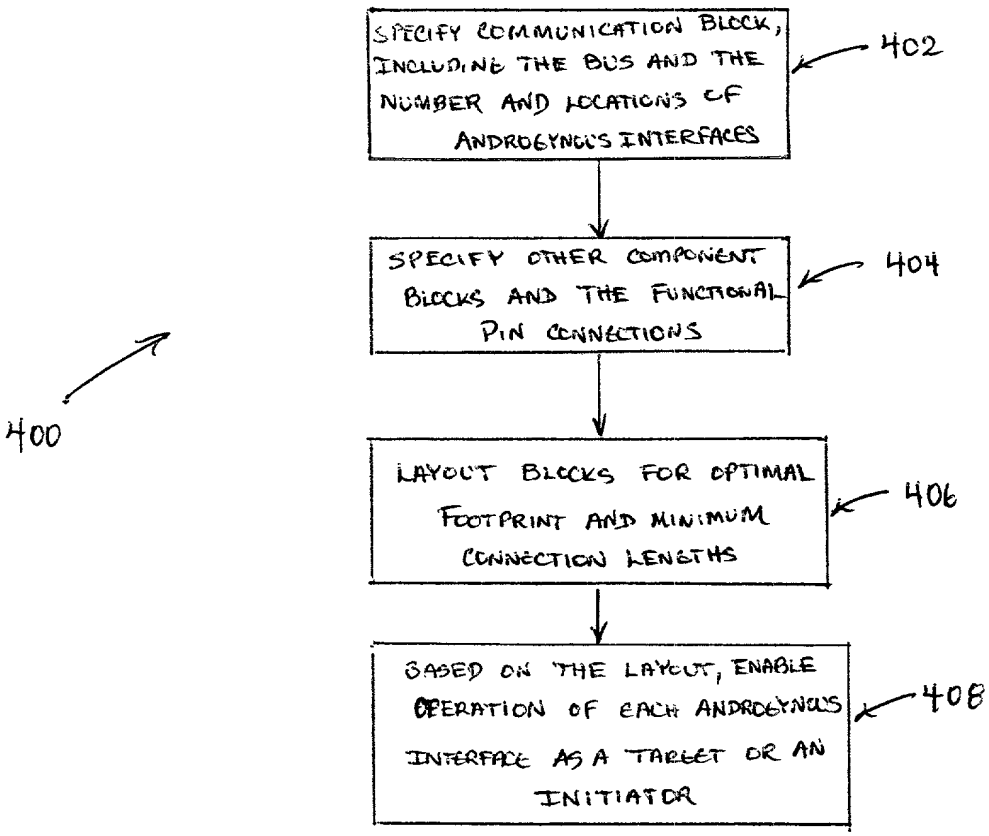

SPECIFY COMMUNICATION BLOCK, INCLUDING THE BUS AND THE NUMBER AND LOCATIONS OF ANDROGYNOUS INTERFACES
402
SPECIFY OTHER COMPONENT BLOCKS AND THE FUNCTIONAL PIN CONNECTIONS
404
400
LAYOUT BLOCKS FOR OPTIMAL FOOTPRINT AND MINIMUM CONNECTION LENGTHS
406
BASED ON THE LAYOUT, ENABLE OPERATION OF EACH ANDROGYNOUS INTERFACE AS A TARGET OR AN INITIATOR
408

FIG. 4

I/T		T/I
R_data	<-	Data
R_addr	<-	Addr
R_cmd	<-	Cmd
R_Plen	<-	Plen
R_Cfix	<-	Cfix
R_Clen	<-	Clen
R_dval	<-	Dval
Ack	->	R_Ack
R_eop	<-	Eop
R_error	<-	Error
Data	->	R_Data
Addr	->	R_Addr
Cmd	->	R_Cmd
Plen	->	R_Plen
Cfix	->	R_Cfix
Clen	->	R_Clen
Dval	->	R_dval
R_Ack	<-	Ack
Eop	->	R_Eop
Error	->	R_Error

FIG. 5

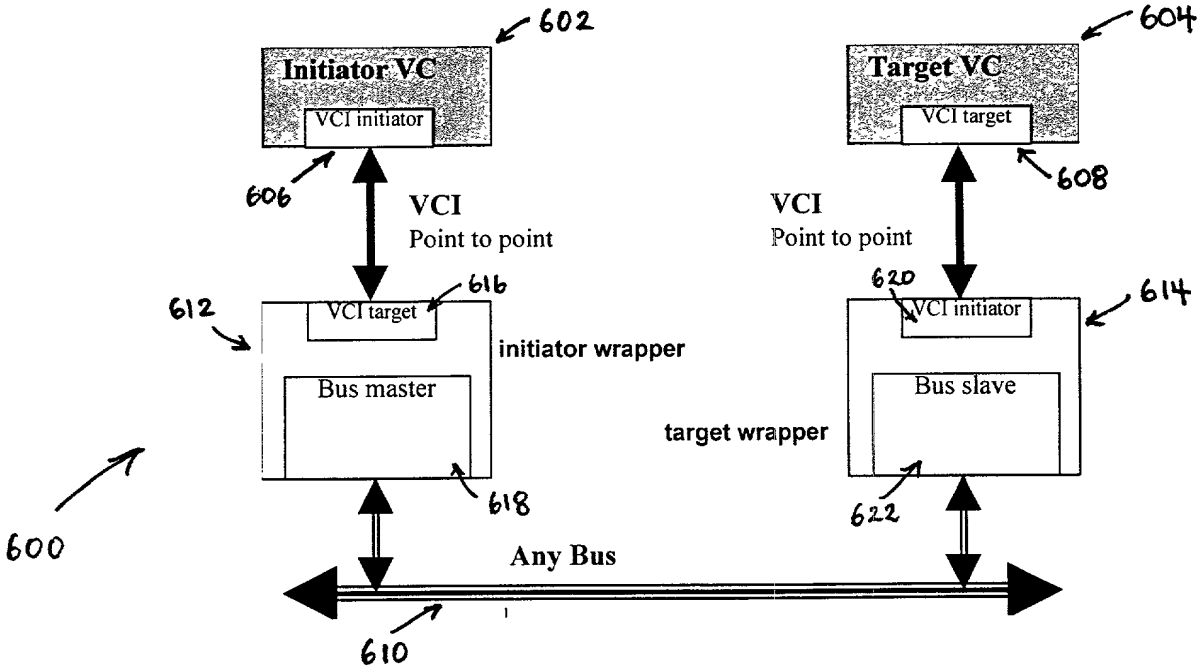

602
Initiator VC
VCI initiator
606
VCI
Point to point
612
VCI target
616
initiator wrapper
Bus master
618
600
Any Bus
610
604
Target VC
VCI target
608
VCI
Point to point
620
VCI initiator
614
target wrapper
Bus slave
622

FIG. 6

SOC Master		VCI Slave	VCI Master		SOC Slave
BCLK	==	CLOCK	CLOCK	==	BCLK
BnRES	==	RESETN	RESETN	==	BnRES
r_gnt	<-	CMDACK	CMDACK	<-	gnt
r_req	->	CMDVAL	CMDVAL	->	req
r_addr	->	ADDR[n-1:0]	ADDR[n-1:0]	->	addr
1111	->	BE[b-1:0|0:b-1]	BE[b-1:0|0:b-1]	->	xxxx
0	->	CFIXED	CFIXED	->	x
0	->	CLEN[q-1:0]	CLEN[q-1:0]	->	x
r_cmd	->	CMD[1:0]	CMD[1:0]	->	cmd
1	->	CONTIG	CONTIG	->	x
r_data	->	WDATA[8b-1:0]	WDATA[8b-1:0]	->	data
r_eop	->	EOP	EOP	->	eop
0	->	CONST	CONST	->	x
r_d_size	->	PLEN[k-1:0]	PLEN[k-1:0]	->	d_size
0	->	WRAP	WRAP	->	x
gnt	->	RSPACK	RSPACK	->	r_gnt
req	<-	RSPVAL	RSPVAL	<-	r_req
data	<-	RDATA [8b-1:0]	RDATA [8b-1:0]	<-	r_data
eop	<-	REOP	REOP	<-	r_eop
error	<-	RERROR	RERROR	<-	r_error

FIG. 7

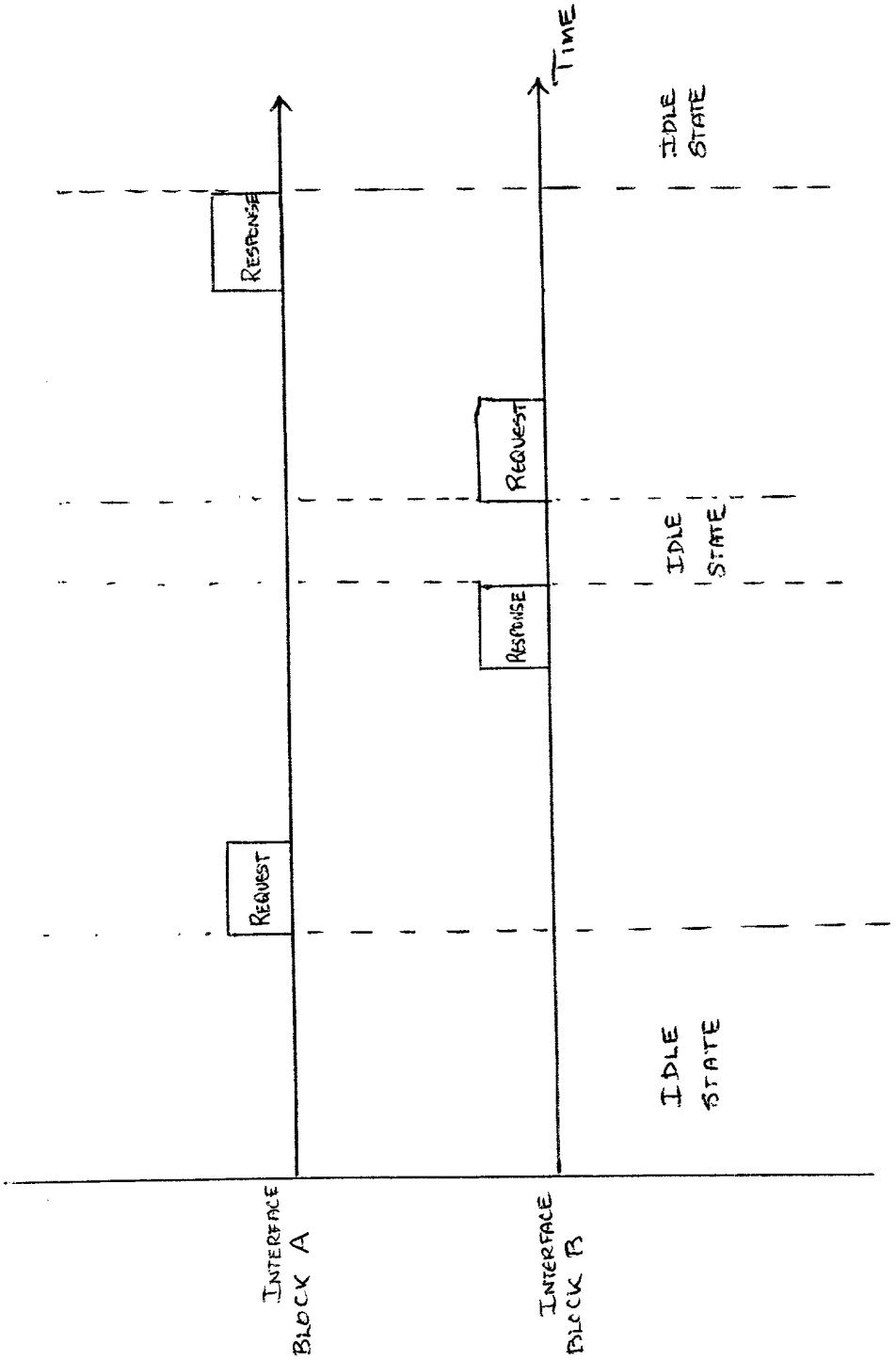

TIME
IDLE STATE
RESPONSE
REQUEST
IDLE STATE
REQUEST
RESPONSE
IDLE STATE
INTERFACE BLOCK A
INTERFACE BLOCK B

FIG. 8